Book review

Review of spreadsheet implementation technology: Basics and extensions, by Peter Sestoft, MIT Press, 2014, ISBN 978-0-262-52664-7
doi:10.1017/S0956796816000204

As functional programmers, we are used to being told that Microsoft Excel is the most popular functional programming platform, with millions of people using the spreadsheet to solve complex scientific and financial problems as a matter of routine. There is literature to support these users, from navigating user interfaces to "tweaking" formulas for efficient re-computation, as well as tightly focussed research communities focussing on user experience and verification. However, until the publication of Sestoft's monograph, there was very little published on engineering aspects of spreadsheets.

This book changes that, and is a splendid introduction to the area that goes into depth about how spreadsheets are implemented, but drawing in also aspects of formal semantics, design choice, history and intellectual property management. The core of the book is an implementation of a spreadsheet in C#, Corecalc, in Chapters 2–5 and its extension to a model with user-defined functions, Funcalc, in Chapters 6–11. Introducing these is the overview "What is a spreadsheet?" which sets the scene. As I suspect many readers would do, I expected this to contain no surprises, but I was wrong. Cell reference formats are introduced early, and the familiar A1 format (letter for column; number for row), and absolute addressing indicated by a $ prefix, is compared with R1C1 indexing, which is invariant when a formula is copied.

Here, in a nutshell, is that makes this such a stimulating read: The prosaic question of representation is embedded in the bigger one of efficiency of what needs to be adjusted when a spreadsheet is used in practice. The choice of R1C1 means that (a) less modification is needed on copying and indeed, (b) representations of formulas can often be shared on copying. These insights come throughout the introduction, and indeed through the whole book; for this audience, it is worth sharing his observation that spreadsheet recalculation is dual to lazy evaluation, with a cell being recalculated when some cell on which it depends is re-calculated: Evaluation is driven by the availability of input rather than (in the case of lazy evaluation) by the demand for output.

The second section covers the C# implementation of Corecalc, with Chapter 2 giving the nuts and bolts details, including classes and method implementations. Whilst it is possible to skip such details, there are also some nuggets to be found, such as using IEEE NaN values, and their payload bits, to transmit different error values during re-calculation, as well as the details how cell move, insertion and deletion are handled. Chapter 3 has a focus on design and engineering, with a detailed comparison of mechanisms to support minimal – or parsimonious – re-calculation, including reading the runes of how it is done in Excel, based on some developer network notes and blog posts. Chapters 4 and 5 cover the support graph: connecting up the links from a cell to all its uses, which underlies the preferred method of recalculation. Whilst the basics are simple, building data structures that can handle realistic-scale data, as well as more complex cell area and array structures, requires a more sophisticated approach.

The final part of the monograph – which brings to the foreground Sestoft's research – describes how user-defined functions can be added to a spreadsheet, building on earlier work by Peyton Jones, Blackwell and Burnett. He shows how functions can be compiled, in JIT

https://doi.org/10.1017/S0956796816000204 Published online by Cambridge University Press

style, into virtual machine code for C#. As has been a feature in earlier chapters, he presents an "aside" on CIL bytecode, so that the reader can follow the exposition without having to go elsewhere. Another aside introduces partial evaluation, and Chapter 10 covers this in the context of the Funcalc spreadsheet.

Overall, this book provides an engaging read for the interested computer scientist who wants to find out more about spreadsheet engineering, but it could also serve as the basis of a challenging advanced undergraduate or postgraduate course in programming language technology. The implementations of Corecalc and Funcalc are freely available under MIT-style licence, and could supplement a course with practical materials. It is underpinned with a comprehensive set of references that put the exposition into its historical context and the writing is lucid and engaging. In summary, I recommend it without reservation.

SIMON THOMPSON
School of Computing, University of Kent, Canterbury, UK

https://doi.org/10.1017/S0956796816000204 Published online by Cambridge University Press